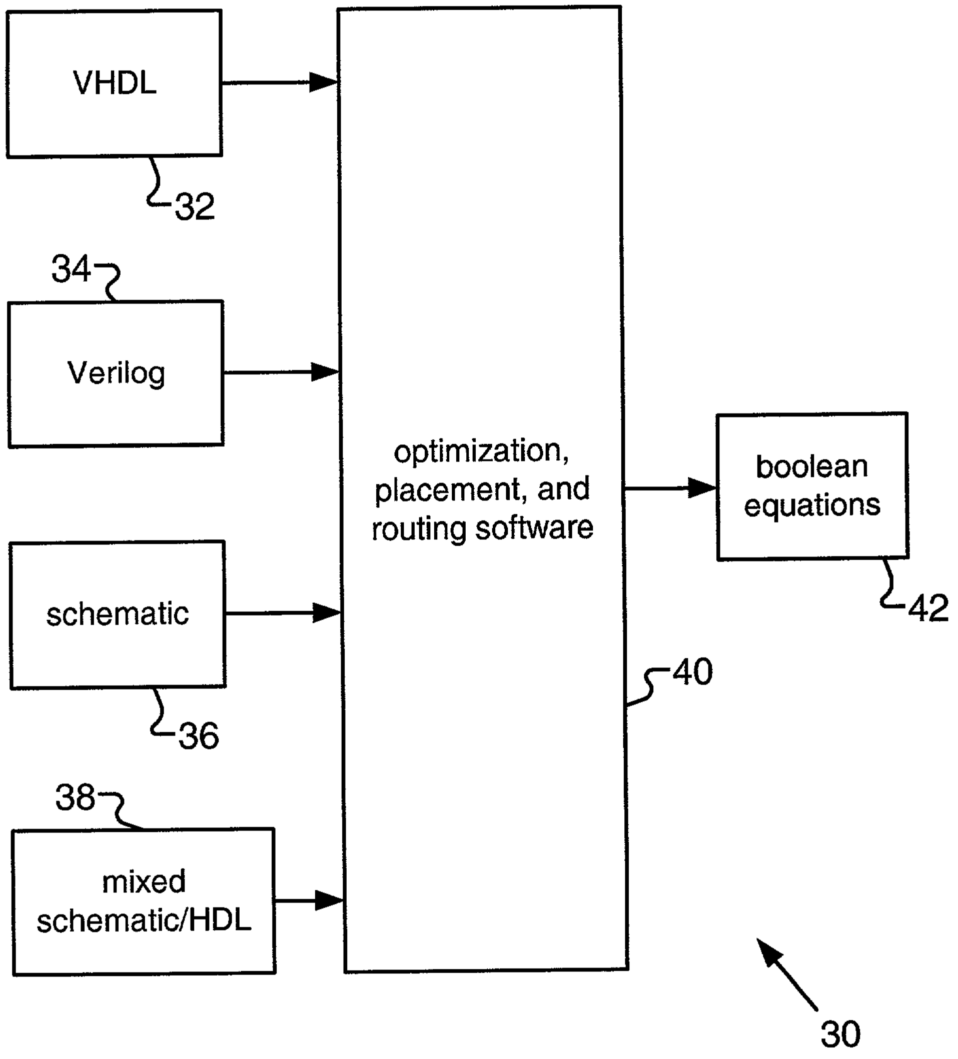

prior art

FIG. 1

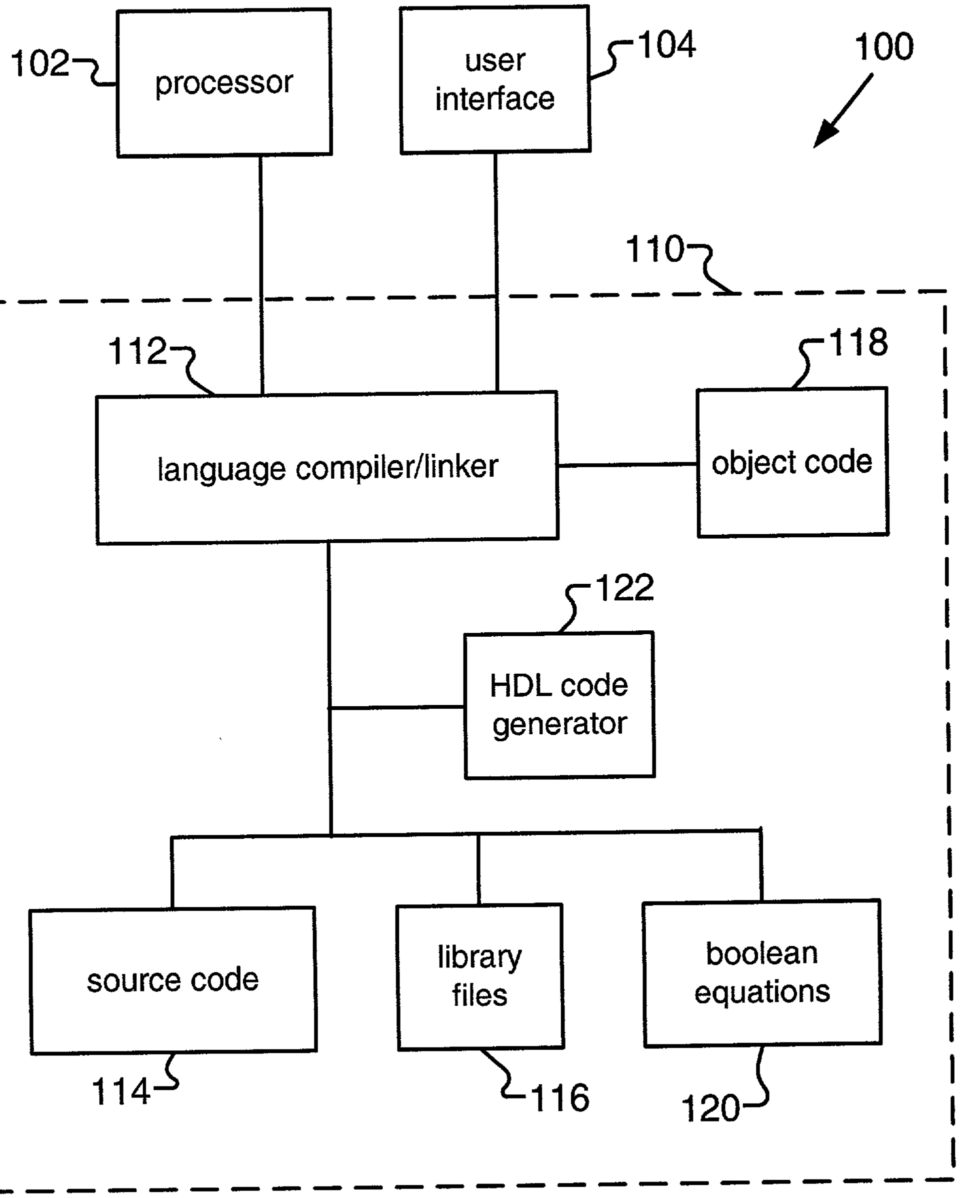
102
processor
user interface
104
100
110
112
language compiler/linker
118
object code
122
HDL code generator
source code
library files
boolean equations
114
116
120

FIG. 2

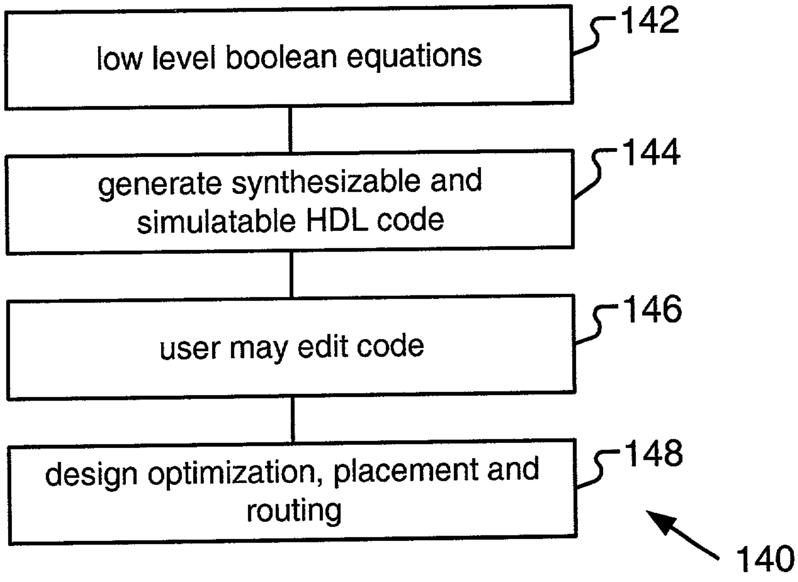
low level boolean equations
142
generate synthesizable and simulatable HDL code
144
user may edit code
146
design optimization, placement and routing
148
140

FIG. 3

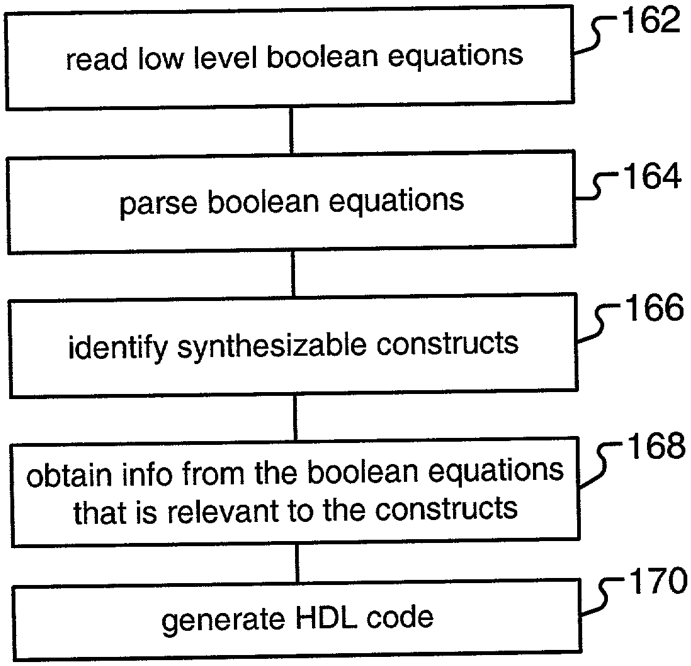
read low level boolean equations
162
parse boolean equations
164
identify synthesizable constructs
166
obtain info from the boolean equations that is relevant to the constructs
168
generate HDL code
170

FIG. 4

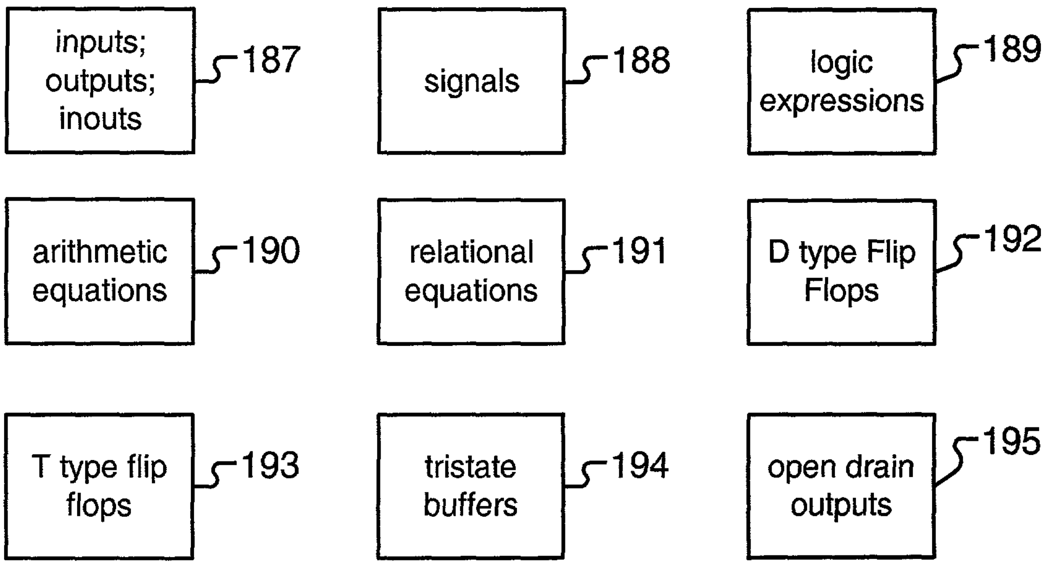
inputs;
outputs;
inouts
187
signals
188
logic
expressions
189
arithmetic
equations
190
relational
equations
191
D type Flip
Flops
192
T type flip
flops
193
tristate
buffers
194
open drain
outputs
195

FIG. 5A

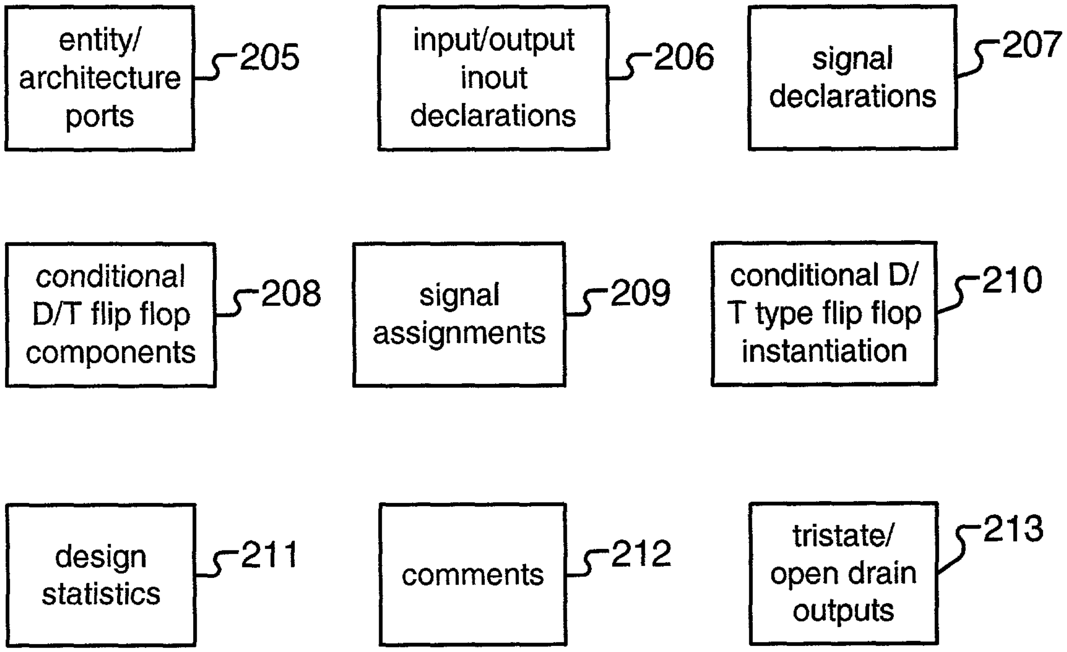
entity/
architecture
ports
205
input/output
inout
declarations
206
signal
declarations
207
conditional
D/T flip flop
components
208
signal
assignments
209
conditional D/
T type flip flop
instantiation
210
design
statistics
211
comments
212
tristate/
open drain
outputs
213

FIG. 6A

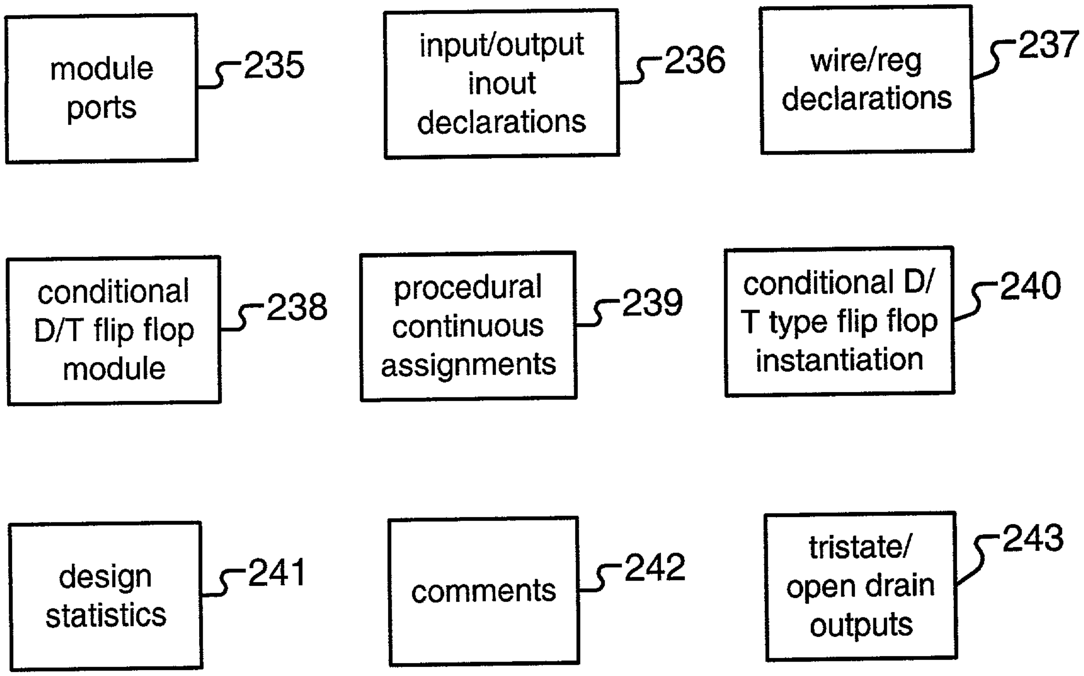
module ports
235
input/output inout declarations
236
wire/reg declarations
237
conditional D/T flip flop module
238
procedural continuous assignments
239
conditional D/ T type flip flop instantiation
240
design statistics
241
comments
242
tristate/ open drain outputs
243

FIG. 6B

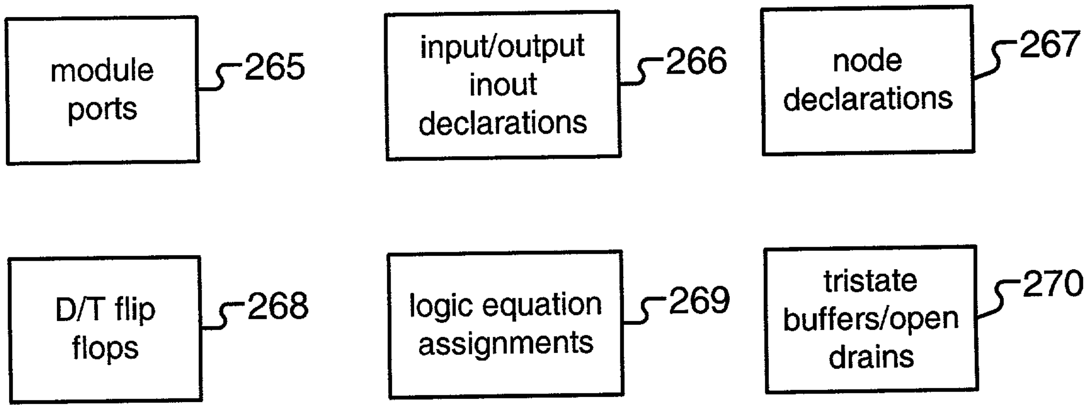
module ports
265
input/output inout declarations
266
node declarations
267
D/T flip flops
268
logic equation assignments
269
tristate buffers/open drains
270

FIG. 6C

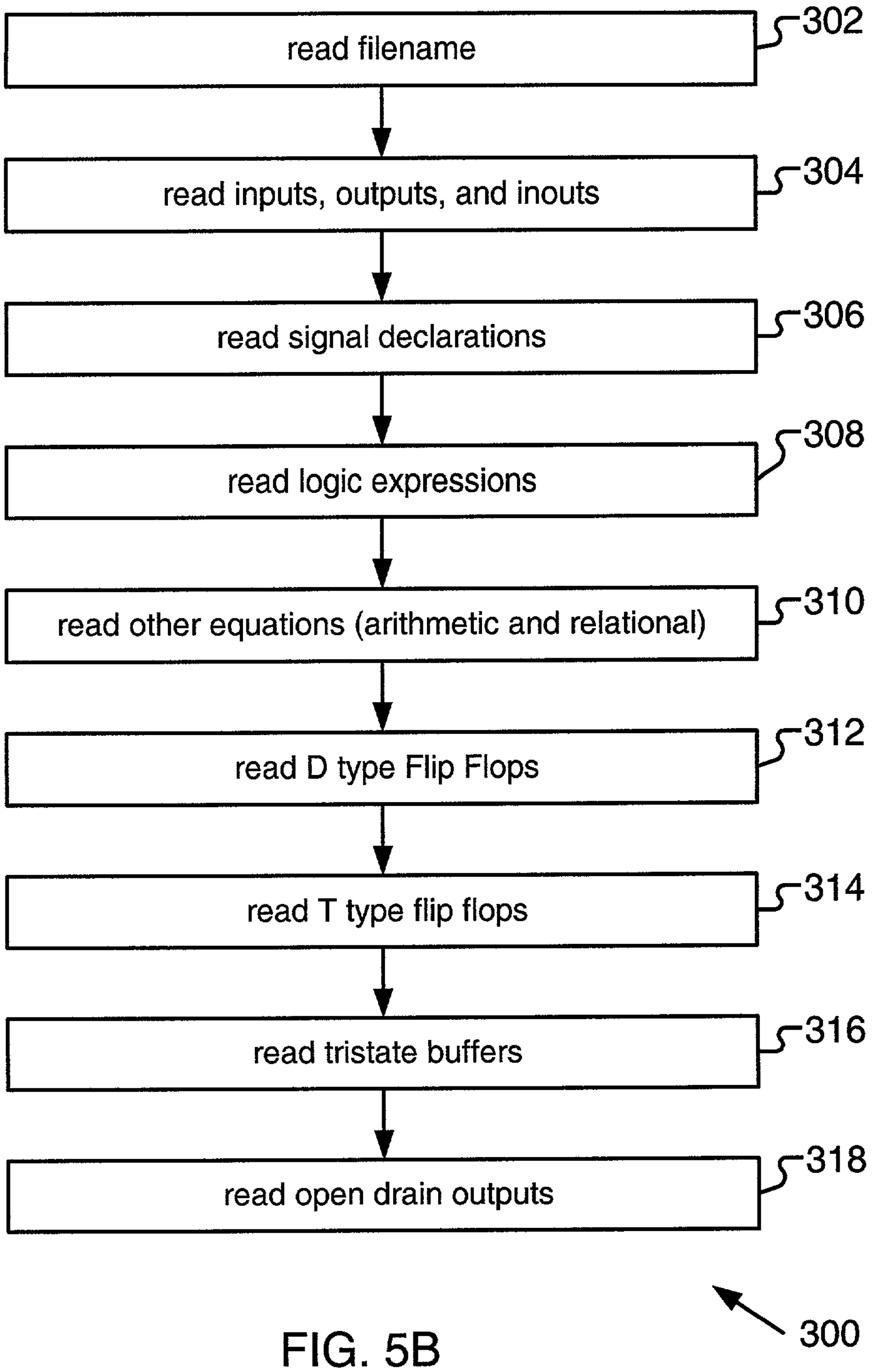
read filename
302
read inputs, outputs, and inouts
304
read signal declarations
306
read logic expressions
308
read other equations (arithmetic and relational)
310
read D type Flip Flops
312
read T type flip flops
314
read tristate buffers
316
read open drain outputs
318
300

FIG. 5B

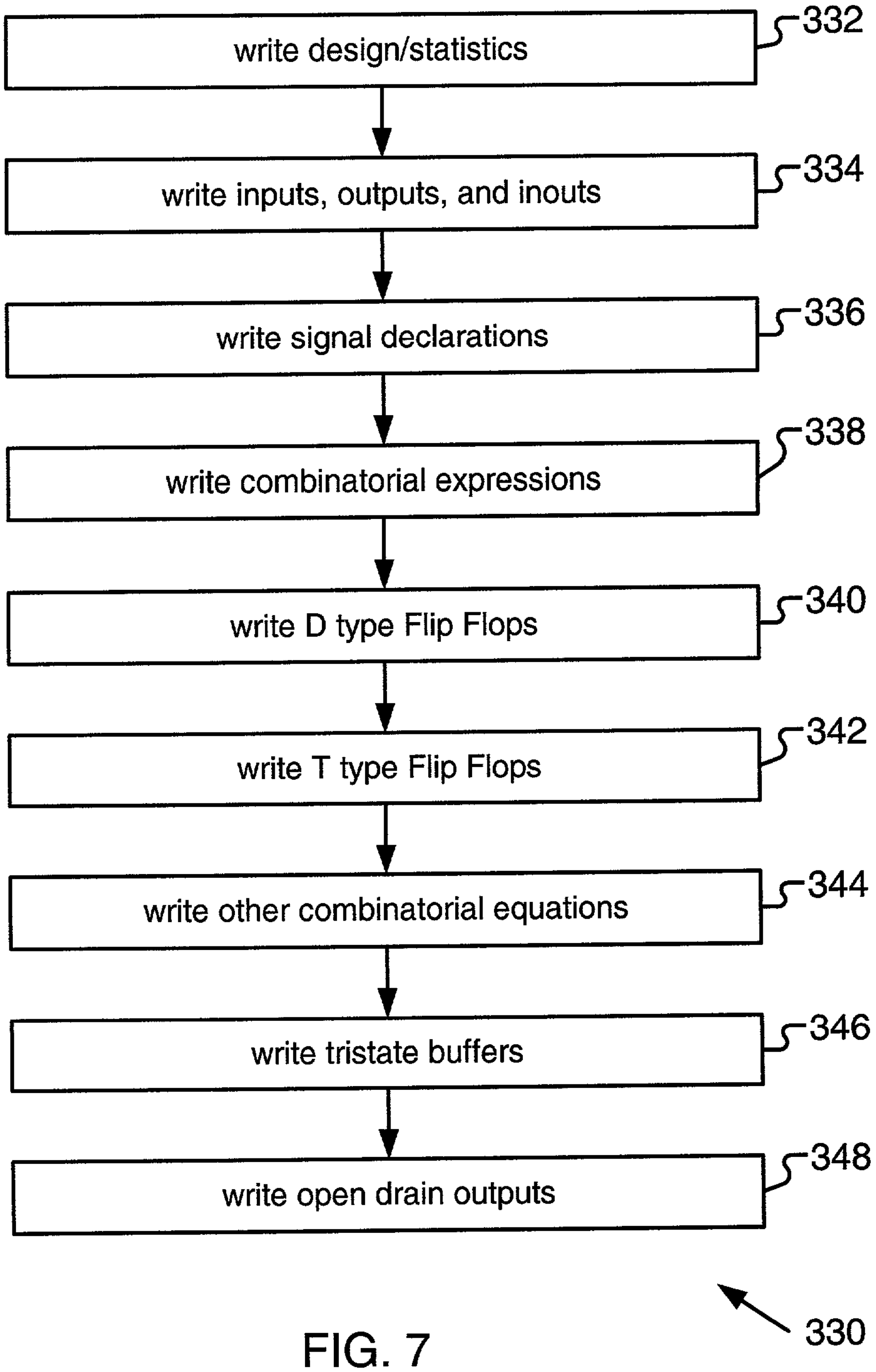
write design/statistics
332
write inputs, outputs, and inouts
334
write signal declarations
336
write combinatorial expressions
338
write D type Flip Flops
340
write T type Flip Flops
342
write other combinatorial equations
344
write tristate buffers
346
write open drain outputs
348
330

FIG. 7

```
Equations

SUBDESIGN 'counter4'
(
    clk                     : INPUT;
    rst                     : INPUT;
    count0                  : OUTPUT;
    count1                  : OUTPUT;
    count2                  : OUTPUT;
    count3                  : OUTPUT;
)

VARIABLE

    _EQ001                  : NODE;
    _EQ002                  : NODE;

BEGIN

 count0  = TFFE( VCC, GLOBAL( clk), !rst,  VCC,  VCC);

 count1  = TFFE( count0, GLOBAL( clk), !rst,  VCC,  VCC);

 count2  = TFFE( _EQ001, GLOBAL( clk), !rst,  VCC,  VCC);
  _EQ001 =  count0 &  count1;

 count3  = TFFE( _EQ002, GLOBAL( clk), !rst,  VCC,  VCC);
  _EQ002 =  count0 &  count1 &  count2;

END;
```

Fig. 8A

```
-- Design name: counter4

-- Design Statistics
-- Number of Inputs       : 2
-- Number of Outs/Inouts  : 4
-- Number of TFFEs        : 4
-- Number of EQ Equations : 2

library ieee;
use ieee.std_logic_1164.all;

entity tffe is
port (q : inout std_logic;
      t : in std_logic;
      clk : in std_logic;
      rst : in std_logic;
      pre : in std_logic;
      ce : in std_logic
);
end tffe;

architecture v1 of tffe is
signal d: std_logic;
begin

process (rst,pre,clk)
begin
if rst = '0' then
q <= '0';
elsif pre = '0' then
q <= '1';
elsif clk'event and clk = '1' then
if ce = '1' then
q <= d ;
end if ;
end if ;
end process ;

d <= t xor q;

end v1;

library ieee;
use ieee.std_logic_1164.all;
```

Fig. 8B-1

```
entity counter4 is
port  (
clk : in std_logic;
rst : in std_logic;
count0 : inout std_logic;
count1 : inout std_logic;
count2 : inout std_logic;
count3 : inout std_logic);
end counter4 ;

architecture conversion of  counter4 is

component tffe
port (q : inout std_logic;
     t : in std_logic;
     clk : in std_logic;
     rst : in std_logic;
     pre : in std_logic;
     ce : in std_logic
);
end component;

signal EQ001 : std_logic;
signal EQ002 : std_logic;

begin

-- 4 TFFE

tffe0 : tffe port map (count0,'1',clk,not  rst,'1','1');
tffe1 : tffe port map (count1,count0,clk,not  rst,'1','1');
tffe2 : tffe port map (count2,EQ001,clk,not  rst,'1','1');
tffe3 : tffe port map (count3,EQ002,clk,not  rst,'1','1');

-- 2 Comb Eqns(s)

EQ001 <= (count0 and  count1);
EQ002 <= (count0 and  count1 and  count2);

-- 0 X Comb Eqn(s)


-- Additional Combinatorial Eqns


-- Assignments for equations w active low LHS


end conversion;
```

Fig. 8B-2

```
/* Design statistics

   Number of Inputs       : 2
   Number of Outputs      : 4
   Number of TFFEs        : 4
   Number of EQ Equations : 2
*/

module dffe (q,d,clk,rst,pre,ce);
output q;
input d,clk,rst,pre,ce;
reg q;

always @(posedge clk or negedge rst or negedge pre)
begin
if (~rst)
q = 1'b0;
else if (~pre)
q = 1'b1;
else if (ce)
q = d;
end

endmodule

module tffe (q,t,clk,rst,pre,ce);
output q;
input t,clk,rst,pre,ce;
wire d;

dffe dffe0 (q,d,clk,rst,pre,ce);

assign d = t ^ q;

endmodule
```

Fig. 8C-1

```
module counter4 (
     clk,
     rst,
     count0,
     count1,
     count2,
     count3);

// 2 Inputs

input clk;
input rst;

// 4 Outputs

output count0;
output count1;
output count2;
output count3;

// 0 reg statements


// 1  wire statements

wire _EQ001;
wire _EQ002;

// 4  TFFE

tffe tffe0 (count0,1'b1,clk,!rst,1'b1,1'b1);
tffe tffe1 (count1,count0,clk,!rst,1'b1,1'b1);
tffe tffe2 (count2,_EQ001,clk,!rst,1'b1,1'b1);
tffe tffe3 (count3,_EQ002,clk,!rst,1'b1,1'b1);

// 2 Comb Eqns(s)

assign _EQ001 = count0 &  count1;
assign _EQ002 = count0 &  count1 &  count2;

// 0 _X Comb Eqns(s)

endmodule
```

Fig. 8C-2

```
" Conversion of counter4.tdo to counter4.abl

   clk                  pin;
    rst                  pin;
    count0               pin;
    count1               pin;
    count2               pin;
    count3               pin;

    _EQ001               NODE;
    _EQ002               NODE;

equations

count0.t = 1 ;
count0.clk = ( clk) ;
count0.ar = !!rst ;
"count0.ap = 1 ;
"count0.ce = 1 ;

count1.t = count0 ;
count1.clk = ( clk) ;
count1.ar = !!rst ;
"count1.ap = 1 ;
"count1.ce = 1 ;

count2.t = _EQ001 ;
count2.clk = ( clk) ;
count2.ar = !!rst ;
"count2.ap = 1 ;
"count2.ce = 1 ;
  _EQ001 =  count0 &  count1;

count3.t = _EQ002 ;
count3.clk = ( clk) ;
count3.ar = !!rst ;
"count3.ap = 1 ;
"count3.ce = 1 ;
  _EQ002 =  count0 &  count1 &  count2;

END;
```

Fig. 8D

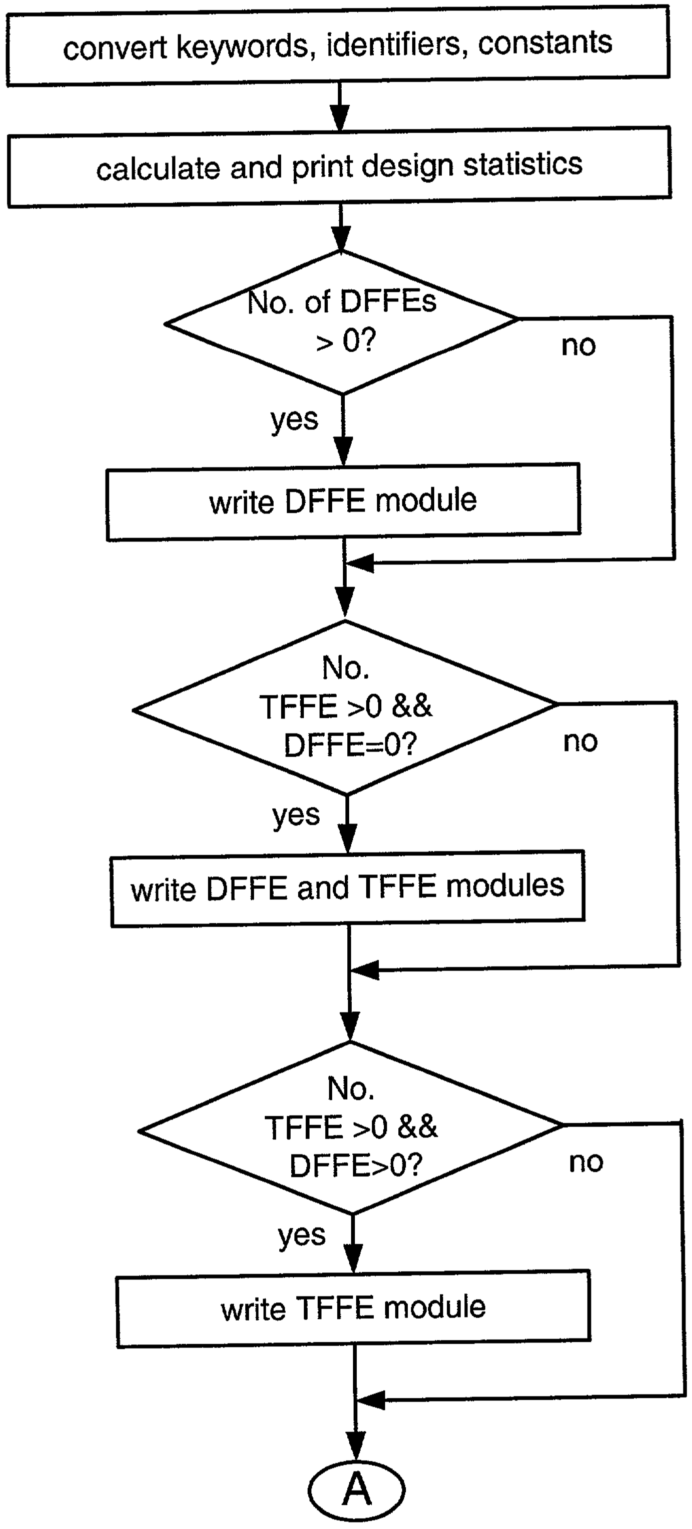
convert keywords, identifiers, constants
calculate and print design statistics
No. of DFFEs > 0?
no
yes
write DFFE module
No. TFFE >0 && DFFE=0?
no
yes
write DFFE and TFFE modules
No. TFFE >0 && DFFE>0?
no
yes
write TFFE module
A

FIG. 9A

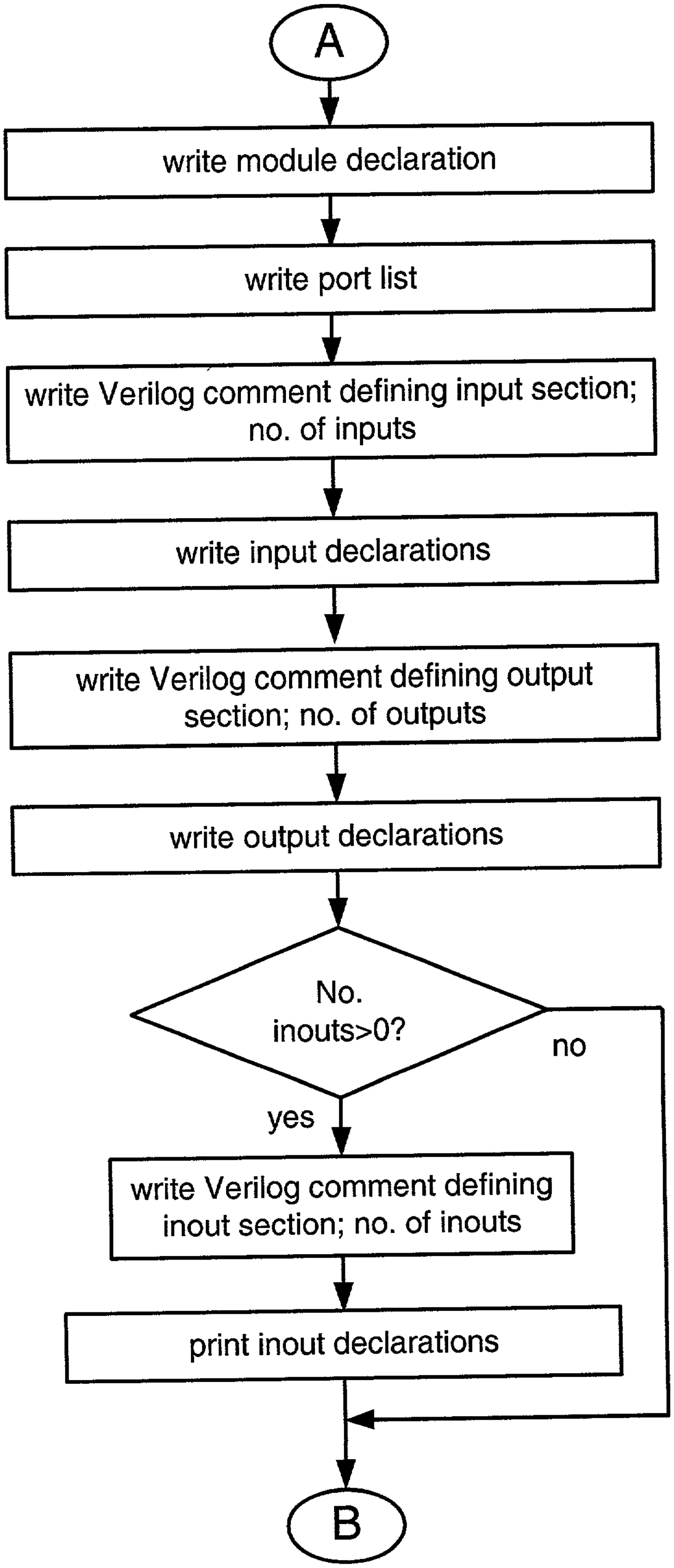
A
write module declaration
write port list
write Verilog comment defining input section; no. of inputs
write input declarations
write Verilog comment defining output section; no. of outputs
write output declarations
No. inouts>0?
no
yes
write Verilog comment defining inout section; no. of inouts
print inout declarations
B

FIG. 9B

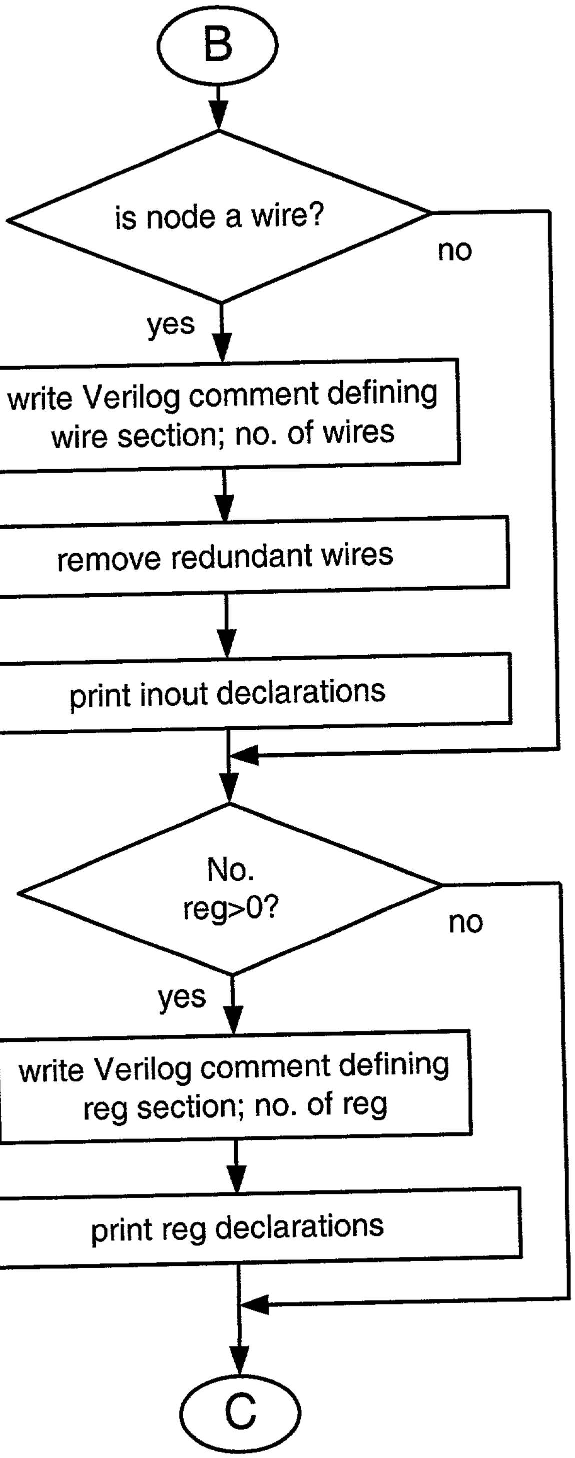
B
is node a wire?
no
yes
write Verilog comment defining wire section; no. of wires
remove redundant wires
print inout declarations
No. reg>0?
no
yes
write Verilog comment defining reg section; no. of reg
print reg declarations
C

FIG. 9C

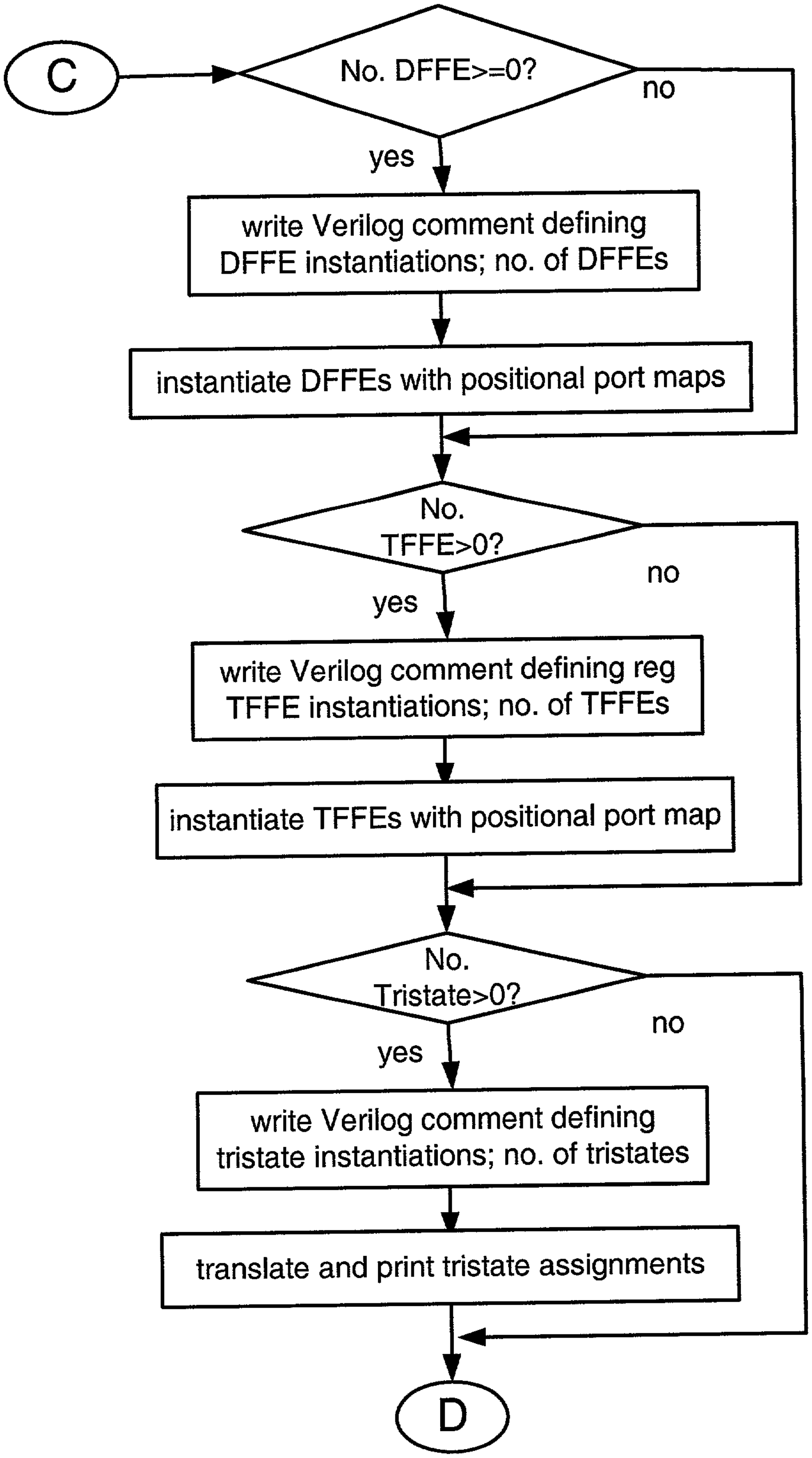
C
No. DFFE>=0?
no
yes
write Verilog comment defining DFFE instantiations; no. of DFFEs
instantiate DFFEs with positional port maps
No. TFFE>0?
no
yes
write Verilog comment defining reg TFFE instantiations; no. of TFFEs
instantiate TFFEs with positional port map
No. Tristate>0?
no
yes
write Verilog comment defining tristate instantiations; no. of tristates
translate and print tristate assignments
D

FIG. 9D

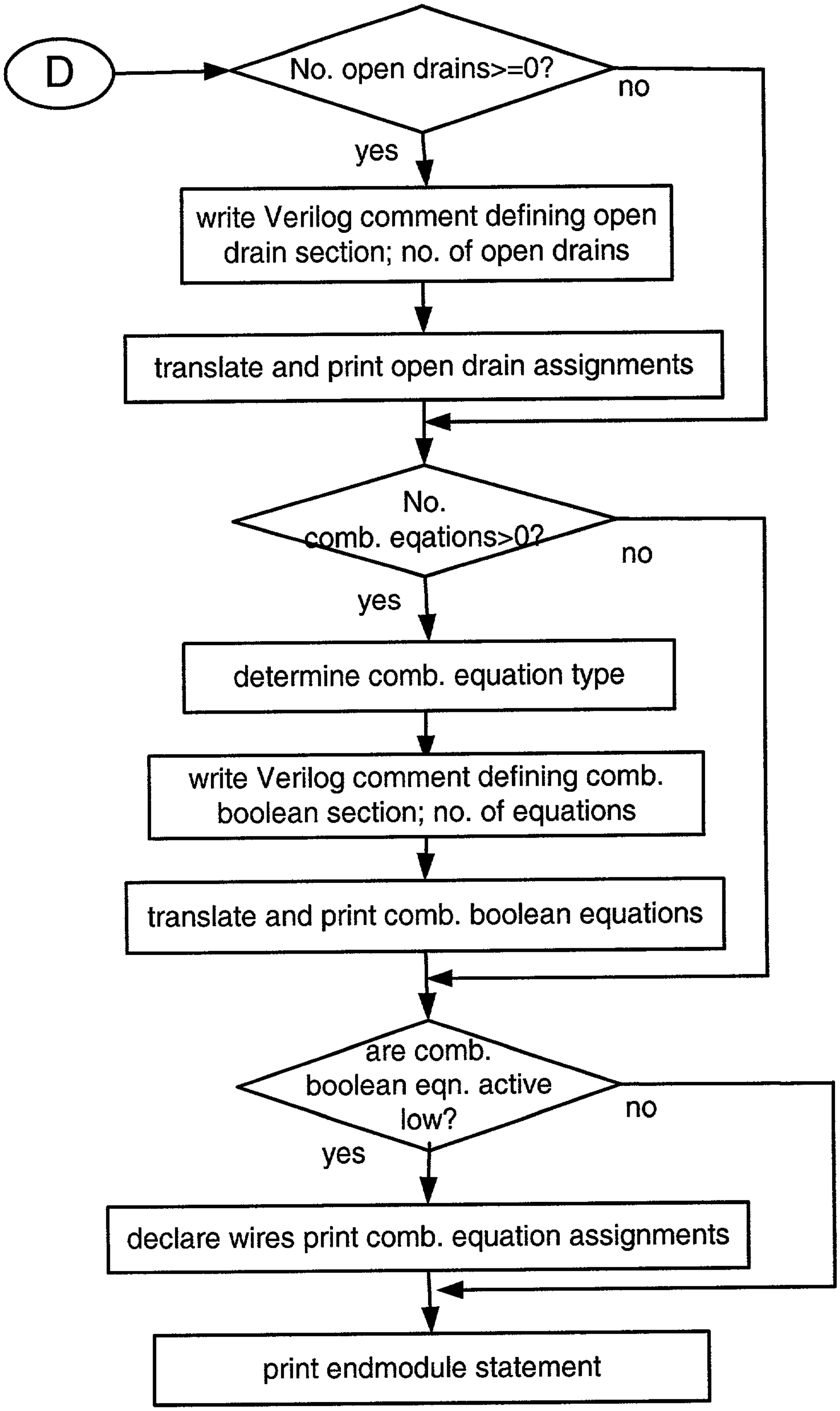
D
No. open drains>=0?
no
yes
write Verilog comment defining open drain section; no. of open drains
translate and print open drain assignments
No. comb. eqations>0?
no
yes
determine comb. equation type
write Verilog comment defining comb. boolean section; no. of equations
translate and print comb. boolean equations
are comb. boolean eqn. active low?
no
yes
declare wires print comb. equation assignments
print endmodule statement

FIG. 9E